A Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures

Manikandasriram Srinivasan Ramanagopal, André Phu-Van Nguyen, and Jerome Le Ny, *Senior Member, IEEE*

***Abstract*— This paper presents a strategy to guide a mobile ground robot equipped with a camera or depth sensor, in order to autonomously map the visible part of a bounded 3-D structure. We describe motion planning algorithms that determine appropriate successive viewpoints and attempt to fill holes automatically in a point cloud produced by the sensing and perception layer. The emphasis is on accurately reconstructing a 3-D model of a structure of moderate size rather than mapping large open environments, with applications for example in architecture, construction, and inspection. The proposed algorithms do not require any initialization in the form of a mesh model or a bounding box, and the paths generated are well adapted to situations where the vision sensor is used simultaneously for mapping and for localizing the robot, in the absence of additional absolute positioning system. We analyze the coverage properties of our policy, and compare its performance with the classic frontier-based exploration algorithm. We illustrate its efficacy for different structure sizes, levels of localization accuracy, and range of the depth sensor, and validate our design on a real-world experiment.**

***Note to Practitioners*—The objective of this paper is to automate the process of building a 3-D model of a structure of interest that is as complete as possible, using a mobile camera or depth sensor, in the absence of any prior information about this structure. Given that increasingly robust solutions for the visual simultaneous localization and mapping problem are now readily available, the key challenge that we address here is to develop motion planning policies to control the trajectory of the sensor in a way that improves the mapping performance. We target in particular scenarios where no external absolute positioning system is available, such as mapping certain indoor environments where GPS signals are blocked. In this case, it is often important to revisit previously seen locations relatively quickly, in order to avoid excessive drift in the dead-reckoning localization system. Our system works by first determining the boundaries of the structure, before attempting to fill the holes in the constructed model. Its performance is illustrated through simulations, and a real-world experiment performed with a depth sensor carried by a mobile manipulator.**

***Index Terms*— Active sensing, active simultaneous localization and mapping (SLAM), autonomous inspection, autonomous mapping, motion planning.**

Manuscript received February 23, 2017; accepted August 16, 2017. This paper was recommended for publication by Associate Editor G. Nejat and Editor Y. Sun upon evaluation of the reviewers' comments. This work was supported in part by NSERC under Grant 435905-13 and in part by the Canada Foundation for Innovation under Grant 32848. The work of M. Srinivasan Ramanagopal was partly done while with Polytechnique Montreal, under a Globalink Fellowship from MITACS. *(Corresponding author: Manikandasriram Srinivasan Ramanagopal.)*

M. Srinivasan Ramanagopal is with the Robotics Institute, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: srmani@umich.edu).

A. P.-V. Nguyen and J. Le Ny are with the Department of Electrical Engineering, Polytechnique Montreal, Montreal, QC H3T 1J4, Canada, and also with GERAD, Montreal, QC H3T 1J4, Canada (e-mail: andre-phu-van.nguyen@polymtl.ca; jerome.le-ny@polymtl.ca).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The Supplementary Material contains a video screen capture that shows the simulated Husky robot following our proposed policies to autonomously map the Small Gamma model. The range of the camera used in this simulation is 4.5m. The video also shows the real world experiment performed indoors using a Husky robot with a Kinova Mico arm carrying a Kinect v2 sensor. This material is 35.8 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2017.2762088

I. Introduction

ACCURATE 3-D computer models of large structures have a wide range of practical applications, from inspecting an aging structure to providing virtual tours of cultural heritage sites [1], [2]. In civil engineering for example, an important problem is that of *construction progress monitoring*, i.e., comparing the state of a building under construction over time to the project plan. The process of regularly updating the estimate of the state of the building has traditionally been performed manually, but in recent years, new methods have been developed to automate it using data obtained from a variety of sensors, e.g., positioning systems, stationary 3-D laser scanners [3], high-resolution video cameras [4], or still cameras carried by unmanned aerial vehicles [5].

This paper considers the problem of guiding in real time a mobile autonomous robot carrying a vision sensor, in order to build a 3-D model of a structure. For this, we need to address two problems. First, we need a robust mapping system that can build the 3-D model in real time when given a sequence of images or depth maps as input. This is a widely researched problem called visual simultaneous localization and mapping (vSLAM) or real-time Structure from Motion (SfM), for which several open source packages offer increasingly accurate and efficient solutions [6], [7]. The second problem relates to active sensing [8], as we need motion planning strategies that can guide a mobile sensor to explore the structure of interest. For mapping, monitoring, or inspection applications, certain classical strategies, such as frontier-based exploration algorithms [9], which guide the robot to previously unexplored regions irrespective of whether it is part of the structure of interest or not, are not necessarily well adapted.

Some recent work considers the problem of reconstructing a 3-D model of arbitrary objects by moving a depth sensor relative to the object using different forms of *next best view* planning algorithms [10], [11]. Typically, these systems iteratively build a complete 3-D model of the object by heuristically choosing the next best viewpoint according to some performance measure. However, much of this paper is restricted to building models of relatively small objects that are bounded by the size of the robot workspace. In contrast, our focus is

on the 3-D reconstruction of larger but still bounded structures such as buildings, which can be several orders of magnitude larger than a mobile robot. The related problem of automated inspection deals with large structures, such as tall buildings [5] and ship hulls. Bircher *et al.* [12] assume that a prior 3-D mesh of the structure to inspect is available and compute a short path connecting viewpoints that together are guaranteed to cover all triangles in the mesh. Englot and Hover [13] begin by assuming a safe bounding box of the hull and construct a coarse mesh of the hull by tracing along the walls of this box in a f xed trajectory without taking feedback from the actual geometry of the structure. Moreover, this coarse mesh is manually processed off ine to yield an accurate 3-D mesh, which is then used to inspect the f ner structural details. Yoder and Scherer [14] also assume a bounding box and develop an algorithm combining next best view planning and frontier-based exploration to encourage coverage of the structure. Sheng *et al.* [15] use a prior CAD model of an aircraft to plan a path for a robotic crawler, such that it inspects all the rivets on the surface of the aircraft. In this paper, however, we do not assume any prior information in terms of a 3-D mesh, CAD model, or a bounding box around the structure, and focus on reactive path planning to build the model online. Our mapping problem is also related to coverage path planning (see [16]–[19] and the references therein), which has traditionally focused on developing algorithms ensuring that a mobile robot passes over all points in a 2-D environment, assuming a suffi iently accurate localization system.

In computer vision and photogrammetry, SfM techniques aim at building a 3-D model of a scene from a large number of images [20]–[23], but much of this paper focuses on batch postprocessing and typically assumes a given data set, whereas here our focus is essentially on how to acquire an appropriate set of images. Let us mention, however, the work of Daftry *et al.* [24], which presents an interactive real-time SfM system providing online feedback to the user taking pictures, alerting him or her when a new picture cannot be properly integrated in the model. Also, Tuite *et al.* [25] develop a competitive game where players are encouraged to take pictures that help build complete 3-D models. We emphasize that we do not discuss in detail the task of actually building a model from a collection of pictures or depth maps, which can be executed by one of the available vSLAM or real-time SfM systems, such as the real-time appearance-based mapping (RTAB-Map) package [6] that we use in our experiments. This paper focuses on actively exploring the environment with an autonomous robot to build a complete model in real time, with our controller taking at any time the current model as an input. Which package we use for model reconstruction has little infl ence on our algorithms, for example, any vSLAM system based on pose-graph optimization [26] could be used. The state-of-the-art batch SfM systems can also be used to postprocess the sequence of images or depth maps captured using our policies in order to obtain a more accurate model off ine. Naturally, eventual completeness of the model is limited by the physical characteristics of the robot, and specif cally the reachable space of the sensor (see Fig. 1).

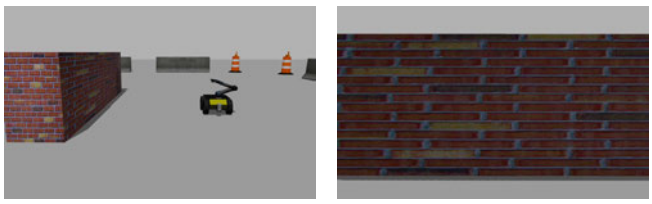
(a)

(b)

Fig. 1. Comparison of (a) simulated model in Gazebo [27] that needs to be mapped and (b) reconstructed 3-D model by a mobile ground robot using our policies. Only the bottom portion is mapped due to the limited reachable space of the sensor.

Finally, another line of work in informative path planning relates to autonomous exploration and coverage of relatively large environments, using variants of frontier-based exploration algorithms [28]–[31]. While these papers focus on path planning to quickly build models of potentially large and complex spaces, they do not address the problem of autonomously delimiting and mapping as completely as possible a specifi bounded structure of interest.

Our contributions can be summarized as follows. After presenting the problem statement in Section II, we develop in Section III a motion planner allowing a ground robot equipped with a camera or depth sensor to autonomously determine the boundaries of an initially unknown structure. Then, Section IV describes an algorithm for detecting missing portions in the 3-D model constructed during the boundary determination phase and an exploration strategy to improve the completeness of the model. In Section V, we analyze the level of coverage completeness that can be expected from our strategy. The behavior of the proposed policies is illustrated in Section VI through simulations, and the resulting accuracy of the constructed models compared with that obtained using the classical frontier-based exploration algorithm. Experimental results are presented in Section VII to validate the algorithms under more realistic illumination conditions. One justificatio for our incremental exploration approach is that we focus on using the vSLAM module both for mapping the structure as well as localizing the robot, although an additional dead-reckoning system, such as wheel odometry, could be present as well. In the absence of an independent source of accurate absolute positioning, it is important to close loops relatively frequently with the vSLAM system, i.e., revisit regions that have already been explored, in order to control the growth of the localization errors building up with visual odometry alone. We also help the vSLAM system by following the boundaries of the structure, where visual features are likely to be present.



(a) (b)

Fig. 2. (a) Starting configuratio for the robot and camera with respect to the structure. (b) Initial image seen by the camera: the robot only knows that the structure in the FOV is the one that should be mapped.

II. Problem Statement and Assumptions

Consider the problem of constructing a 3-D model of a given structure of f nite size, e.g., a monument or a building, using a mobile ground robot carrying an imaging or depth sensor, such as a Kinect, a monocular or stereo camera, or an LIDAR. Initially, no approximate model of the structure nor map of the environment is available, and the actual size of the structure is also unknown. The sensor (also called camera in the following) provides a sequence of point clouds obtained directly or computed from depth and/or luminance images. These local point clouds, together with an estimate of the sensor trajectory, can then be assembled and registered in a coordinate frame in real time using available SLAM algorithms, such as RTAB-Map [6] or RGBD-SLAM [7], and postprocessing then allows us to build a dense 3-D model or a 3-D occupancy grid stored in an OctoMap [32]. We do not directly address here the model reconstruction problem in vSLAM. Instead, we focus on determining good trajectories for the robot allowing a vSLAM module (and potentially a batch SfM module in postprocessing) to produce a high-quality model, which ideally should capture the entire visible portion of the structure accurately. A key challenge is to develop strategies that are applicable for any type of structure while respecting the physical limitations of the platform.

We assume that initially, the robot is positioned along the structure to be mapped, with the camera capturing point clouds mounted on its right and facing the structure at a distance D measured in a horizontal plane (see Fig. 2). In normal operations, we wish to maintain this distance D between the structure and the path of the camera, where D is chosen based on the camera's resolution. Def ne a global f xed Frame of Reference (FoR) $\mathsf{G} := \{O_\mathsf{g}, \mathbf{x}_\mathsf{g}, \mathbf{y}_\mathsf{g}, \mathbf{z}_\mathsf{g}\}$, in which the global point cloud is to be assembled. Note that we write vectors in bold. The robot FoR $\mathsf{R} := \{O_\mathsf{r}, \mathbf{x}_\mathsf{r}, \mathbf{y}_\mathsf{r}, \mathbf{z}_\mathsf{r}\}$ (forward, left, up) coincides initially with G, but is attached to a point O_r that moves along with the robot. For concreteness to describe our scenario and algorithms, the camera FoR $\mathsf{C} := \{O_\mathsf{c}, \mathbf{x}_\mathsf{c}, \mathbf{y}_\mathsf{c}, \mathbf{z}_\mathsf{c}\}$ is assumed to be rigidly attached to the robot except for the yaw motion, which is left unconstrained.

Assumption 1: The center O_c of the camera mounted on the mobile ground robot has f xed coordinates $(0, 0, h_\mathsf{c})$ in frame R, and in addition, we always maintain $\mathbf{z}_\mathsf{r} = \mathbf{z}_\mathsf{c}$.

Such a choice of camera conf guration determines which parts of the structure are not visible at all, and hence cannot be mapped by any algorithm implemented on this platform. However, other system conf gurations could be handled with some of the more generic tools developed in this paper.

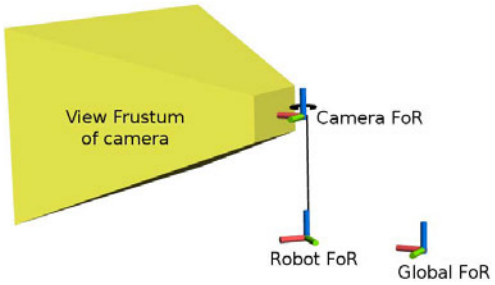


Fig. 3. Camera is kept at a constant height above the robot's base. The red, green, and blue lines correspond to the x-, y-, and z-axes, respectively, and both the camera and the robot can rotate along their z-axes. The yellow region corresponds to the view frustum of the camera.

Fig. 3 shows our conventions for the different FoRs used. The imaging plane of the camera is def ned by $\mathbf{y}_\mathsf{c}\mathbf{z}_\mathsf{c}$, with $\mathbf{x}_\mathsf{c}$ pointing toward the front of the camera on the optical axis. Coordinates in the camera, robot, and global FoR are denoted using superscripts as $\mathbf{v}^\mathsf{c}$, $\mathbf{v}^\mathsf{r}$, and $\mathbf{v}^\mathsf{g}$, respectively, for a vector $\mathbf{v}$. We assume that initially, $\mathbf{x}_\mathsf{c} = -\mathbf{y}_\mathsf{r}$ so that the camera points to the right of the robot. We make two additional assumptions for simplicity of exposition. The f rst one guarantees that there exist collision free paths around the structure.

Assumption 2: The horizontal distance of the closest obstacle from the structure is at least 2 D.

The next assumption simplifie the problem of detecting, tracking, and removing the ground surface from point clouds, a processing step performed in Algorithm 1 to compute waypoints that only depend on the structure to inspect.

Assumption 3: The structure and the robot are placed on the horizontal plane $z^\mathsf{g} = 0$. In particular, we have $\mathbf{z}_\mathsf{c} = \mathbf{z}_\mathsf{r} = \mathbf{z}_\mathsf{g}$.

In the following, we fi the z-coordinate of O_r to be zero. A consequence of these assumptions is that relatively horizontal surfaces that are at the same height or above the camera center, for example, cannot be mapped, and the maximum height (measured in the R or G frame) of the structure that can be mapped is $H_{\max} = h_\mathsf{c} + D \tan \psi/2$, where ψ is the vertical angle of view of the camera. Assumption 3 could be removed by using recent classificatio systems that can differentiate between ground and nonground regions [33] to preprocess the point clouds before sending them to our system.

Finally, there are additional implicit assumptions that we state informally. First, since we rely on an external mapping module to build the 3-D model, the conditions that allow this module to operate suff ciently reliably must be met. For example, vSLAM generally requires appropriate scene illumination and the presence of a suff ciently rich set of visual features. Second, we concentrate on the reconstruction of the details of the model at a scale comparable with or larger than the typical length of the robot. If features at a smaller scale need to be included, e.g., f ne structural details on a wall, our system could be augmented with a more local planner for a robotic arm carrying the sensor [10], [34], as well as targeted computer vision techniques [23]. Finally, for reasons explained in Section III-C, we assume that the robot is equipped with sensors capable of detecting obstacles in a 180° region ahead of it and within a distance of D (see Fig. 8).

We divide our mapping process into two phases (see Fig. 4). The f rst is the perimeter exploration (PE) phase, during which

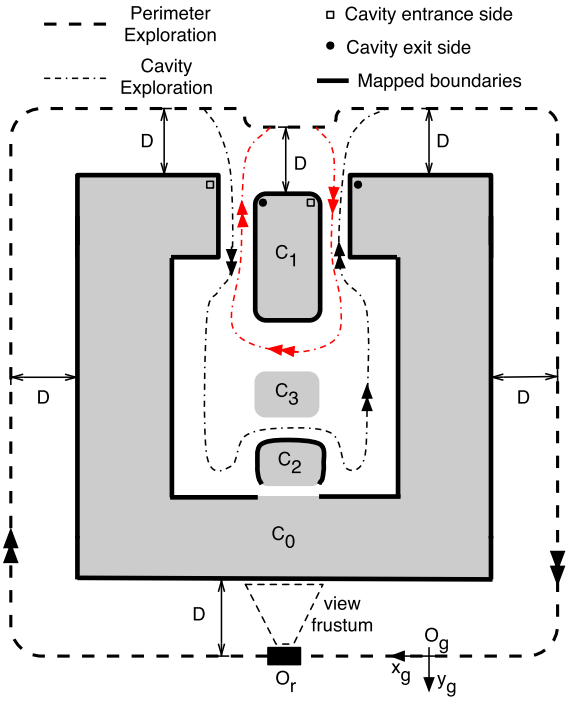


Fig. 4. Overview of the two phases of the mapping strategy. The gray area represents the slice $\mathcal{M}$ in the plane $z^g = h_c$ of the structure to map, with the assumption that any potential hanging structure above the white area leaves enough vertical clearance for the mobile ground robot to navigate.

the robot moves with the structure on its right to determine its boundaries. The robot continuously moves toward previously unseen regions of the structure, with the exploration directed toward finding the limits of the structure and closing a first loop around it relatively quickly rather than trying to map all its details. The PE phase ends when our algorithm detects that the robot has returned to the neighborhood of its starting point $\mathsf{O_g}$, and the vSLAM module detects a global loop closure. After completing the PE phase, the system determines the locations of potential missing parts in the constructed 3-D model. Next, in the cavity exploration (CE) phase, the system explores these missing parts in the model. Sections III and IV explain each step of our process in detail.

III. Perimeter Exploration

In this section, we present a method to autonomously determine the boundaries of an unknown structure. From Assumptions 2 and 3, $z^{\mathsf{g}} = 0$ and $z^{\mathsf{g}} = H_{\max}$ are bounding horizontal planes for the model. The remaining problem is to determine the expansion of the structure in the $\mathbf{x_g y_g}$ plane. To do this, the robot moves clockwise around the structure by determining online a discrete sequence of successive goals or waypoints. It tries to keep the optical axis of the depth sensor approximately perpendicular to the structure, which maximizes the depth resolution at which a given portion of the structure is captured, and increases the density of captured points. It also tries to maintain the camera center O_c on a smooth path at a fixed distance D from the structure.

A. Determination of the Next Goal

The pseudocode to determine the next position and orientation of the camera in our PE algorithm is shown in

Algorithm 1: Algorithm for Computing the Next Goal for the Camera Using the Current Point Cloud in the Camera FoR

1: **function** COMPUTENEXTGOAL(cloud_full)
2: cloud ← PCLremoveGroundPlane(cloud_full)
3: cloud_slice ← f lterForwardSlice(cloud)
4: $\overline{p}^{\mathsf{c}} \leftarrow$ PCLcompute3Dcentroid(cloud_slice)
5: $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3; \lambda_1, \lambda_2, \lambda_3] \leftarrow$ PCA(cloud_slice)
6: $\tilde{\mathbf{n}} \leftarrow \mathbf{v}_3 - (\mathbf{v}_3 \cdot \mathbf{z_c})\mathbf{v}_3$ ▷ Projection on the $\mathbf{x_c y_c}$ plane
7: $\mathbf{n} \leftarrow \tilde{\mathbf{n}}\ \mathrm{sign}(\tilde{\mathbf{n}} \cdot \overrightarrow{O_c \overline{p}^{\mathsf{c}}})$; $\mathbf{n} \leftarrow \mathbf{n}/\|\mathbf{n}\|$
8: $\mathbf{r} \leftarrow \mathbf{z_c} \times \mathbf{n}$
9: $goal \leftarrow \overline{p}^{\mathsf{c}} - D\,\mathbf{n} + step\,\mathbf{r}$
10: **return** $goal$, $\mathbf{n}$
11: **end function**

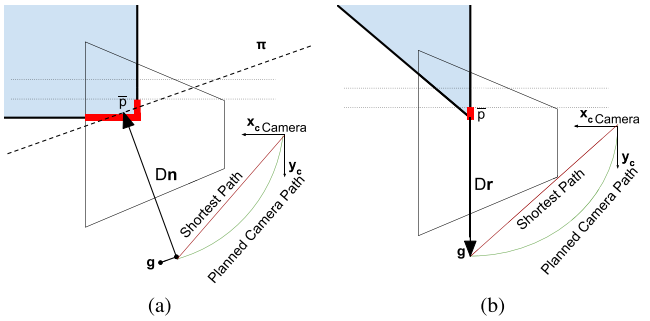


Fig. 5. Top–down view illustrating the computation of the next goal, for a corner section of the structure. (a) Forward slice $\mathcal{S}$ (highlighted in red) contains a portion of the farther section of the corner. (b) For the acute corner, it does not and becomes very narrow.

Algorithm 1. It takes as input the current point cloud produced by the camera in its FoR. For its implementation, we rely on the point cloud library (PCL) [35].

Since the next goal should depend only on the structure, we first remove the ground plane from the captured point cloud by removing all points below a certain height to obtain a point cloud $\mathcal{P}$. Next, on line 3, we select a subset $\mathcal{S}$ of $\mathcal{P}$ referred to as the forward slice, which adjoins the part of the structure that must be explored next [see Fig. 5(a)]. Concretely, we choose $\mathcal{S}$ so that its y^{c}-coordinates satisfy $y^{\mathsf{c}}_{\max} - (y^{\mathsf{c}}_{\max} - y^{\mathsf{c}}_{\min}/3) \leq y^{\mathsf{c}} \leq y^{\mathsf{c}}_{\max}$, where $y^{\mathsf{c}}_{\min}$ and $y^{\mathsf{c}}_{\max}$ are the minimum and maximum y^{c}-coordinate values for all points in $\mathcal{P}$. On line 5, following [36], we compute via principal component analysis the normal direction to that plane Π, which best fits $\mathcal{S}$. In more detail, denote $\mathcal{S} = \{p^{\mathsf{c}}_i : i = 1, 2, \ldots, m\}$ and define the covariance matrix $\mathbf{X} = (1/m)\sum_{i=1}^{m}(p^{\mathsf{c}}_i - \overline{p}^{\mathsf{c}})(p^{\mathsf{c}}_i - \overline{p}^{\mathsf{c}})^T$, where $\overline{p}^{\mathsf{c}} = (1/m)\sum_{i=1}^{m} p^{\mathsf{c}}_i$ is the centroid of $\mathcal{S}$ computed on line 4. We compute the eigenvectors $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ of $\mathbf{X}$, ordered here by decreasing value of the eigenvalues λ_1, λ_2, and λ_3. The eigenvector $\mathbf{v}_3$ for the smallest eigenvalue corresponds to the normal to the plane Π.

The algorithm returns $\mathbf{n}$, computed from the projection of the normal vector $\mathbf{v}_3$ on the $\mathbf{x_c y_c}$ plane, and taken to point in the direction of the vector $\overrightarrow{O_c \overline{p}^{\mathsf{c}}}$. This vector $\mathbf{n}$ defines the desired orientation of the camera. The algorithm also returns the next goal point $goal = \overline{p}^{\mathsf{c}} - D\mathbf{n} + step\,\mathbf{r}$ for the center O_c of the camera, where $\mathbf{r} = \mathbf{z_c} \times \mathbf{n}$ is computed on line 8, and $step = (y^{\mathsf{c}}_{\max} - y^{\mathsf{c}}_{\min}/6)$. The term $step\,\mathbf{r}$, which is along

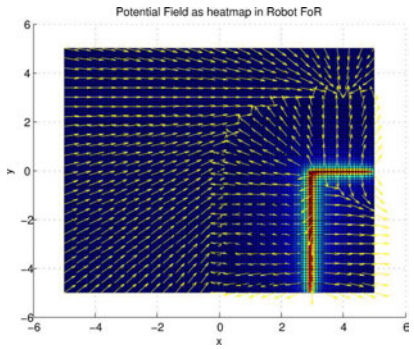


Fig. 6. Potential field for a goal at $(4, 3)$ with $D = 3$ is shown as a heat map and the corresponding gradient vectors are shown as a vector field

the plane Π, is used to shift the goal forward so that both sections of a corner fall in the field of view (FOV) of the camera, as in the situation shown in Fig. 5(a). This prevents the algorithm from making slow progress around corners. Furthermore, the interior angle of a corner could be acute, as shown in Fig. 5(b), and consequently, the farther section of the corner would not be visible from the camera. Such a case can be detected by monitoring the width of $\mathcal{S}$ to fall below a threshold. In this case, we modify the computation of the goal to be $goal = \overline{p}^{\mathsf{c}} + D\mathbf{r}$, which allows the robot to move around sharp corners of the structure. Finally, the computed camera pose is transformed into the global FoR to obtain the next goal point g^{g} for the camera center O_{c}. We simplify the notation g^{g} to g in the following, where we work in the global reference frame.

B. Local Path Planning to the Next Goal

In order to move the camera center to g while keeping it approximately at the desired distance D from the structure along the way, we use a local path planner based on potential fields [37], [38]. A potential function encoding the structure as obstacles in the neighborhood of the camera, as well as the goal g, is sampled in the form of a cost map on a local 2-D grid of size $2D \times 2D$ centered on the camera's current position (see Fig. 6). Assumption 2 guarantees that all the occupied cells in this cost map denote the structure itself. For k occupied cells centered at $\{x_j\}_{j=1}^{k}$, the potential function $N(x)$ is defined as

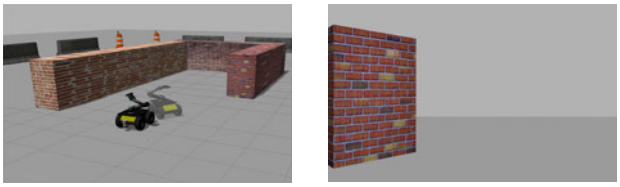
$$N(x) = \alpha \|x - g\|^2 + \sum_{j=1}^{k} I_j(x) d_j(x) \tag{1}$$

with

$$d_j(x) = \frac{1}{\beta \|x - x_j\|}; \quad I_j(x) = \begin{cases} 1, & \text{if } \|x - x_j\| \leq D \\ 0, & \text{otherwise} \end{cases}$$

for some scalar parameters α and β. Here, d_j is the repulsion from the jth occupied cell, and is limited by I_j to a neighborhood of radius D around the cell. A path for the camera is obtained by following the negative gradient of N, i.e., $\dot{x} = -\nabla N(x)$. Denoting $J_x = \{j : I_j(x) = 1\}$ the occupied cells in the D-neighborhood of x, we have

$$-\nabla N(x) = 2\alpha(g - x) + \sum_{i \in J_x} \frac{1}{\beta \|x - x_i\|^3}(x - x_i). \tag{2}$$



(a) (b)

Fig. 7. (a) While the robot is following the structure, its forward facing sensors detect an obstacle ahead (robot configuratio shown in faded colors). (b) Obstacle is outside the FOV of the camera. The position of the robot at the next waypoint along the new direction to explore, determined by using the arm to scan ahead, is shown in bright colors.

Let $\mathcal{M}_D = \{x : J_x \neq \emptyset\}$ denote the region that is at distance at most D from the structure. Assuming a small value of β, the summation term in (2) is dominant whenever $x \in \mathcal{M}_D$ and pushes the path away from the structure. However, this term vanishes as soon as $x \notin \mathcal{M}_D$. Then, assuming that the camera starts at x_0 on the boundary $\partial\mathcal{M}_D$ of $\mathcal{M}_D$, it remains approximately on $\partial\mathcal{M}_D$ if $\overrightarrow{xg}$ points toward the interior of $\mathcal{M}_D$. It is possible that this condition is not satisfied by the point g computed in Section III-A, in which case we replace g by g_1, which is obtained by selecting a new $goal = \overline{p}^{\mathsf{c}} - D'\,\mathbf{n} + step\,\mathbf{r}$ for $D' < D$, such that this condition is satisfie . The path will then slide on $\partial\mathcal{M}_D$ until it reaches its goal [39]. Finally, this path for the center of the camera is used to compute a corresponding path for the center of the robot, which then needs to be tracked using a platform specifi controller.

Overall, during the PE phase, the robot attempts to maintain a viewpoint orthogonal to the structure, even though it replans for a new goal according to Algorithm 1 only at discrete times. Note that only the computation of the next goal happens at discrete instants but the vSLAM module updates the model at a higher rate as per the capabilities of the hardware.

C. Replanning Due to the Structure Interferring

Assumption 2 guarantees that the robot can move sufficiently freely around the structure, but this does not prevent the structure itself from interfering with the path planned above. Consider the situation shown in Fig. 7(a). The wall ahead of the robot does not fall into the FOV of the camera due to the limited horizontal angle of view, yet the robot should not approach this wall closer than a distance D. Hence, if the robot detects obstacles in its forward D-neighborhood, it is stopped at its current position and the yaw motion of the camera is used to scan ahead and face the new section of the structure. More precisely, as shown in Fig. 8, we use the costmap from Section III-B to turn the camera to face along the direction from the robot center O_{r} to the first occupied cell in the D-neighborhood of the robot. The next goal is then recomputed using the newly captured point cloud.

D. End of the PE Phase

The end of the PE phase corresponds to the robot closing a loop around the structure. Therefore, we require that the vSLAM module detects a global loop closure based on the captured images, i.e., recognizes that the robot has returned to

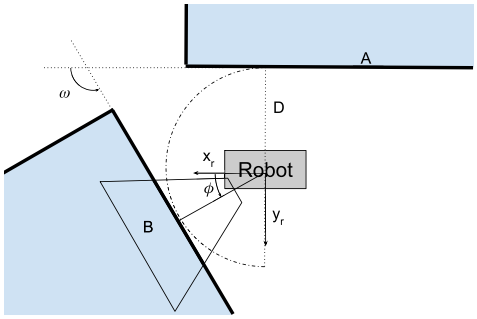


Fig. 8. When the robot is currently following Section A of the structure, a later Section B of the structure could interfere with the planned path. The angle ω made by Section B with respect to Section A satisfies $\omega \in [0, \pi)$. Also, the two sections could be connected to form a nonconvex corner.

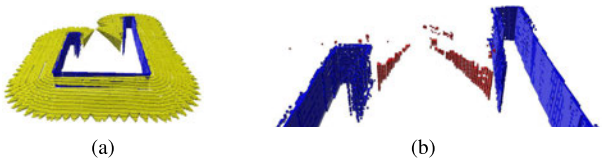
the vicinity of a known point. The robot continues traveling on the PE path until this condition is met. Detecting a global loop closure is not necessarily straightforward because of localization errors, notably the drift accumulating in dead-reckoning systems such as the visual odometry function of the vSLAM module, or the wheel odometry system. However, it is typically possible to place a unique object or mark on or near the structure in the initial FOV, which helps prevent incorrect loop closures. If available, absolute positioning sensors such as a GPS receiver in the case of outdoor operations can also indirectly help improve the loop closure detection by limiting the localization drift. One can also use the measurements of a compass to detect when the robot is traveling along an edge of the structure that has the same orientation as the starting edge, and focus the search for a loop closure along these edges.

IV. Completing the Model: Cavity Exploration

There are two possible types of flaws in the model obtained at the end of the PE phase. Type I flaws correspond to holes that are present in the already explored regions. As noted in Section II, these holes could be due to limitations of the sensor or local occlusions caused by small irregularities in the structure itself, and should be filled using a platform with a more appropriate reachable space, and hence, we do not consider them further. Type II flaws, called cavities in the following, correspond to regions that were skipped during the PE phase, due to the situation depicted in Fig. 7 in particular. These cavities will be filled during the CE phase, where the robot is allowed to move closer to the structure, although this means that the model will not necessarily be reconstructed up to a height $H_{\max}$ in some places.

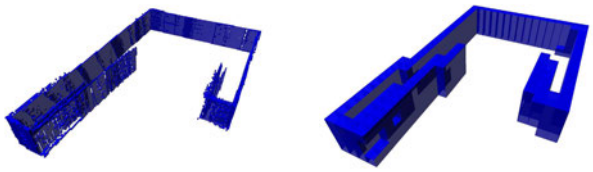
A. Cavity Entrances

In this section, we describe an algorithm to determine the locations of the entrances of the cavities in the model, which will be subsequently used by the CE strategy. We use a voxel-based 3-D occupancy grid constructed from the global point cloud, and maintained in a hierarchical tree data structure by the OctoMap [32] library. Internally, this library performs ray casting operations, labeling the occupancy measurement of each voxel along the line segment from the camera position to each point in the point cloud as *free* and the point itself as



(a) (b)

Fig. 9. (a) Constructed OctoMap with occupied voxels shown in blue and frontier voxels shown in yellow. These yellow voxels form the boundary of the explored region, but most of them lie along the top and bottom faces of the view frustums. (b) Cavity entrance voxels are shown in red.



(a) (b)

Fig. 10. OctoMap queried at depth level 16 and 13, respectively. In this paper, the value of D is 3 m, and the threshold d_0 is chosen as 0.4 m. (a) Leaf size: 0.05 m and depth: 16. (b) Leaf size: 0.4 m and depth: 13.

occupied. For this, we require the vSLAM module to provide the sequence of point clouds and associated estimated camera positions used in assembling the current model. All voxels in the occupancy grid that are not labeled free or occupied are called *unknown*. Using the constructed OctoMap, we compute a set of *frontier* voxels, whose definition is adapted from [9].

Definition 1: A frontier voxel is a free voxel with at least one neighboring unknown voxel.

Recall that the camera is constrained to move in a horizontal plane during the PE phase. Consequently, many frontier voxels lie along the top and bottom faces of the view frustums [see Fig. 9(a)] but do not correspond to cavities to explore. We can ignore them by only considering frontier voxels for which the normal vector $\mathbf{n}$, computed using the nearby frontier voxels [36], makes a sufficiently small angle with the horizontal plane. In other words, we keep only the frontier voxels for which the z-coordinate of the normal $\mathbf{n}$ satisfies $|n_z^g| < \alpha$, for some chosen threshold α. Next, Type I flaws can result in frontier voxels, which we also want to exclude from consideration. Therefore, we require that the distance to the closest occupied voxel should be greater than some threshold d_0, which can be chosen as a small fraction of the distance maintained from the structure, say $0.1D$. As the number of voxels in a typical structure is very large, we do not perform this thresholding exactly, but instead, we use an estimate for the distance to the structure obtained from OctoMap. The hierarchical structure of OctoMap allows efficient multiresolution queries (see Fig. 10), and thus, we keep as cavity entrance voxels only those that are marked free at a resolution of approximately d_0.

Finally, we call *cavity entrance voxels* the frontier voxels that satisfy the two preceding conditions [see Fig. 9(b)]. The cavity entrance voxels are clustered using an Euclidean clustering algorithm from PCL [35], and each cluster is referred to as a *cavity entrance*. Moreover, there could be some sparsely located cavity entrance voxels, which are removed by setting a minimum size for the cavity entrance.

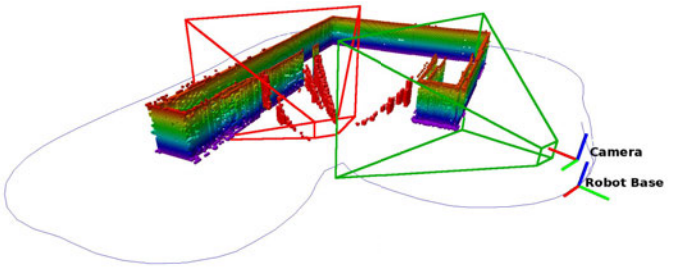


Fig. 11. Robot at a starting viewpoint for exploring a cavity. All the cavity entrance voxels are also shown. We also show on the left an ending viewpoint, where the vSLAM system detects that it is back in a region already explored during the PE phase.

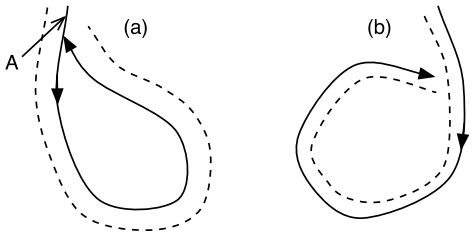


Fig. 12. Impossibility of self-intersection during PE. Dashed curve: boundary of the structure. Solid curve: path of the robot.

B. Cavity Exploration

Once the cavity entrances have been determined, we can start the CE phase. We explore each detected cavity using a motion analogous to the PE phase, wherein we maintain the structure to the right at a distance $\Delta \in [\delta, D]$ that is determined online based on the available clearance in the cavity and δ is the minimum required distance for the mapping module. For this, we require a starting viewpoint for each cavity entrance and an algorithm to compute Δ. The starting viewpoint is chosen from the set of camera poses returned by the vSLAM module during the PE phase, such that the centroid of the cavity entrance lies within the view frustum. Additionally, the centroid should not be occluded by the structure from the camera position. From these camera poses, the one with the earliest timestamp is chosen as starting viewpoint (see Fig. 11).

The timestamps of the starting viewpoints of the cavity entrances are used to sort them in increasing order, and each of the cavities is explored in sequence. A typical cavity has at least two cavity entrances bordering it, as shown in Fig. 9(b), and it is possible to have more cavity entrances in some cases. During the CE phase, if the centroid of a cavity entrance falls within the view frustum of the current camera position and is not occluded by the structure, we remove that cavity entrance from our list.

Exploring conf ned regions during the CE phase requires certain modification to the PE policy. Recall that the system skipped the cavities during the PE phase as the robot came closer than a distance D from the structure. Therefore, during the CE phase, only the region directly ahead of the robot and within a distance Δ is checked for interference of the computed path with the structure. Moreover, our potential fiel -based local path planner now returns paths that maintain a distance Δ from the structure. For this, using the notation of Sections III-A and III-B, we modify *goal* as $goal \leftarrow \overline{p}^{\mathsf{c}} - \Delta\,\mathbf{n} + step\,\mathbf{r}$ and transform to the global FoR to obtain the new point g. The distance Δ is chosen by starting from the minimum value δ and increasing it until we reach a local minimum of $N_\Delta(g)$ along $\mathbf{n}$, where the defi ition of N_Δ is adapted from (1) with Δ replacing D. Similarly, when an acute angled corner is encountered during the CE phase, we modify *goal* as $goal \leftarrow \overline{p}^{\mathsf{c}} + \Delta\,\mathbf{r}$.

When the robot exits a cavity, the point clouds captured by the camera correspond to parts of the structure that are already present in the model from the PE phase. Consequently, the system can detect that it has f nished exploring the current cavity by monitoring the loop closures obtained by the vSLAM module. The robot can then choose the next region to explore from its current list of remaining cavity entrances, and can travel there by following again the PE path. Alternatively, the number of changes in the occupancy measurements of the OctoMap could be used to detect the end of the cavity, as point clouds captured after exiting the cavity ideally would not add new information to the OctoMap. But this solution tends to be less robust, because localization errors and sensor noise can induce a large number of changes even when the camera is viewing a region that is already present in the model.

V. Coverage Analysis

In this section, we provide some analysis of the coverage completeness of the PE and CE strategies. To simplify the discussion, we focus on the case of simple structures consisting of vertical walls, potentially supporting hanging structures under which the mobile robot is able to pass. We then analyze the boundary coverage in 2-D for the slice $\mathcal{M}$ of the structure on the plane $z^{\mathsf{g}} = h_{\mathsf{c}}$ (see Fig. 4).

First, we analyze the PE phase. We assume that the path planner is able to keep the robot at distance D from $\mathcal{M}$, and in other words, the robot's path remains on the boundary $\partial\mathcal{M}_D$ def ned in Section III-B, keeping the structure on its right. Note that $\mathcal{M}_D$ is the Minkowski sum $\mathcal{M} \oplus \mathcal{B}_D$ of $\mathcal{M}$ and a closed disk of radius D.

Lemma 1: The path followed by the robot during PE phase cannot self-intersect, except at the initial point O_{g}.

Proof: During the PE phase, the robot keeps the structure at distance D on its right as it moves forward. Fig. 12 then shows the impossibility for the robot's path to intersect itself during PE. Indeed, in Fig. 12(a) of a counterclockwise cycle, one can show that before the merging point, the robot's obstacle detector would have seen the structure on its left at distance at most D (point A in Fig. 12), and implemented the left turn as explained in Section III-C. In Fig. 12(b) of a clockwise cycle, the tube of width $2D$ around the robot's trajectory would collide with the structure before closing the path, which again would have induced a left turn. In both cases, we have a contradiction. ■

Recall that a simple closed curve (SCC) is a nonself-intersecting, continuous loop. We then have the following.

Corollary 1: Suppose $\partial\mathcal{M}_D$ consists of a f nite set of disjoint SCCs. Then, during the PE phase, the robot travels on the SCC of $\partial\mathcal{M}_D$ on which it initially started, in the

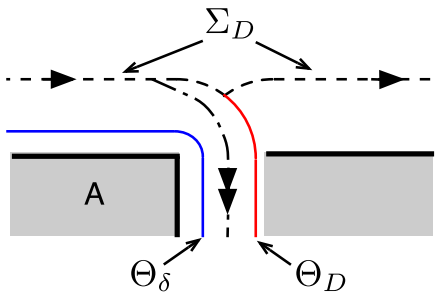


Fig. 13. Illustration of the notation used in Proposition 1. Here, $\mathcal{C}_\delta = A \oplus \mathcal{B}_\delta$.

direction that keeps $\mathcal{M}_D$ on its right. Moreover, assuming the loop closure detection does not incorrectly terminate the PE phase too early, the robot reaches back its starting point O_9 on this curve.

Proof: The robot progresses along $\partial\mathcal{M}_D$, and its path cannot self-intersect by Lemma 1, so it must eventually reach back its starting point, since the length of $\partial\mathcal{M}_D$ is fi ite. It cannot switch to another SCC than the one on which it started, since it would violate the assumption that the planner maintains a distance D with $\mathcal{M}$. ■

Corollary 1 characterizes the part of the boundary of $\mathcal{M}_D$ that the PE strategy covers, assuming the path planner and PE termination algorithm work correctly. The robot ideally travels on an SCC that is part of $\partial\mathcal{M}_D$, which we call the *PE curve* in the following. We orient this curve in the direction of travel of the robot, with $\mathcal{M}_D$ on the right.

Let us now turn to the analysis of the CE phase. Let $\mathcal{M}_\delta = \mathcal{M} \oplus \mathcal{B}_\delta$ be the Minkowski sum of $\mathcal{M}$ and the closed disk of radius δ, where δ is the minimum horizontal clearance define in Section IV-B. We work under the mild assumption that both $\partial\mathcal{M}_D$ and $\partial\mathcal{M}_\delta$ consist of a fi ite set of disjoint SCCs, although $\partial\mathcal{M}_D$ can have a strictly smaller number of such curves in general. The notation of the following proposition is illustrated in Fig. 13.

Proposition 1: Let Σ_D be the oriented PE curve, and $\mathcal{C}_\delta$ be a connected component of $\mathcal{M}_\delta$ in the region on the right of Σ_D. Let Θ_δ be one of the SCC forming $\partial\mathcal{C}_\delta$, orient Θ_δ, such that C_δ is on its right, and let Θ_D be the (possibly empty) SCC forming the boundary of $\Theta_\delta \oplus \mathcal{B}_{D-\delta}$ on the left of Θ_δ. If $\Theta_D \cap \Sigma_D \neq \emptyset$, then at the end of the PE and the CE phase, the view frustum has covered the curve Θ_δ.

Referring to Fig. 4, $\mathcal{M}$ has four components C_i, $i = 0, \ldots, 3$. $\mathcal{M}_D$ has a unique component, since by adding a buffer D, the components merge into one. Note, however, that in general, $\mathcal{M}_D$ does not have to be simply connected, nor even path connected. The dashed line representing the PE curve is also the boundary of $\mathcal{M}_D$. Now, $\mathcal{M}_\delta$ has three components, because C_0 and C_2 merge once we add a buffer δ. C_1 and C_3 remain disconnected in $\mathcal{M}_\delta$ however, which allows the robot to enter the passages separating C_0 and C_1 on the one hand, and C_2 and C_3 on the other hand. At the end of CE, the boundary of the components $C_1 \oplus \mathcal{B}_\delta$ and $(C_0 \cup C_2) \oplus \mathcal{B}_\delta$ will be mapped, but $C_3 \oplus \mathcal{B}_\delta$ does not satisfy the hypothesis of Proposition 1 (the boundary of $C_3 \oplus \mathcal{B}_D$ does not share any point with the PE curve), and in this case, its boundary indeed is not mapped. The robot cannot map the whole boundary of C_0 or C_2 individually, since it cannot pass between these two structures that are less than δ apart.

Proof: Note that Θ_D is an SCC that forms part or all of ∂C_D, where $C_D = C_\delta \oplus \mathcal{B}_{D-\delta}$, i.e., Θ_D consists of points that are at distance D of the portion of the structure in C_δ. As a result, all the points belonging to Θ_D must be either also on Σ_D or on the right of Σ_D. A fi st possibility is that $\Theta_D = \Sigma_D$, in which case the curve Θ_δ is covered at the end of the PE phase.

If Θ_δ is not covered at the end of the PE phase, there is a point on Θ_δ that lies on a cavity entrance (frontier boundary between the free and unknown regions) and that is reachable by a path starting from Σ_D (since C_δ is a connected component of $\mathcal{M}_\delta$, a robot could travel along Θ_δ during the CE phase). Assuming this point is detected by the procedure of Section IV-A, during the CE phase, the robot will travel to this point and remove it from its list of cavities to explore, keeping C_δ on its right along the way. It will then continue following a path along C_δ contained in the annulus between Θ_δ and Θ_D, until Θ_δ has been entirely covered by the view frustum. The coverage of Θ_δ terminates, since it is an SSC. ■

In conclusion, the cavity entrances computed at the end of the PE phase act as attractors for the robot during the CE phase. However, the robot only covers those frontier voxels that it can reach while still keeping the structure on its right as a guide and remaining at a distance between δ and D away from it. For example, if it enters a large room after going through a cavity entrance, it will not try to cover the area far away from the walls (hence, it does not try to cover obstacle C_3 in Fig. 4). One could potentially attempt to cover these interior areas as well at the same time, e.g., by using a 2-D coverage algorithm when we enter a wide cavity, but this would require, in general, a suff ciently precise absolute positioning system complementing the odometry information of the vSLAM module. This might not be a trivial requirement, for example, because the structure itself might obstruct GPS reception. Instead, our algorithm is motivated by the fact that keeping the structure in range helps maintain the accuracy of the visual odometry component of the vSLAM module. By trying to exit a cavity quickly once we enter it, the vSLAM module can also close loops more frequently as the robot returns to the PE curve, before accumulating too much error through the odometry.

VI. Simulation Results

We illustrate the behavior of our policies via 3-D simulations for different sizes of the structure, camera range values, and localization accuracy levels for the robot. The implementation of our motion planning policies is integrated with the robot operating system (ROS) navigation stack [40], which is supported by many mobile ground robots. All the simulations are performed using the Gazebo simulator [27]. The vSLAM algorithm used is RTAB-Map [6].

The simulations are carried out with publicly available models of a Clearpath Husky A200 robot and a Kinect depth sensor whose range can be varied [41] (see Fig. 2).

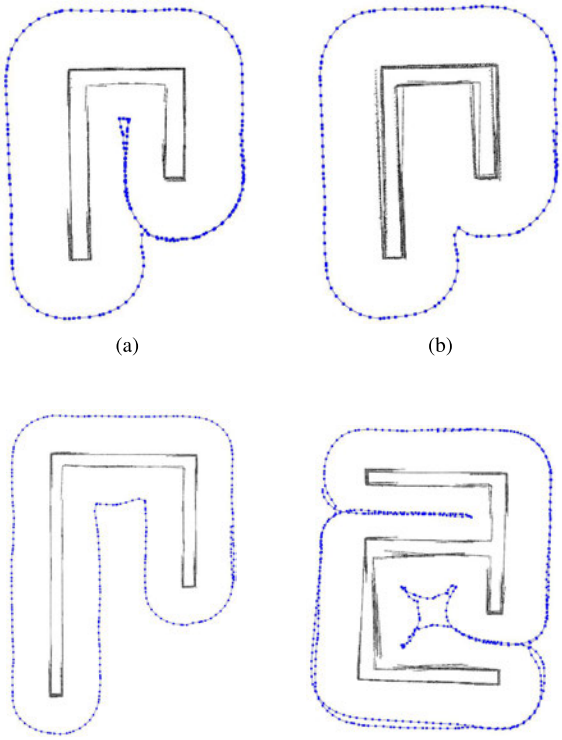


Fig. 14. Projection of the reconstructed model on the $\mathbf{x_g y_g}$ plane is shown in black line, and the trajectory followed by the robot based on our policies is shown in blue line. (a) Model: small Γ and range: 4.5 m. (b) Model: small Γ and range: 12 m. (c) Model: large Γ and range: 4.5 m. (d) Model: large a and range: 4.5 m.

TABLE I

SIMULATION RESULTS FOR DIFFERENT SIZES OF THE STRUCTURE AND RANGE OF THE CAMERA

Model	Perimeter	Camera Range	Path Length
Small Γ	42m	4.5m	72.08m
Small Γ	42m	12.0m	53.79m
Large Γ	84m	4.5m	106.23m
Large a	94m	4.5m	179.65m

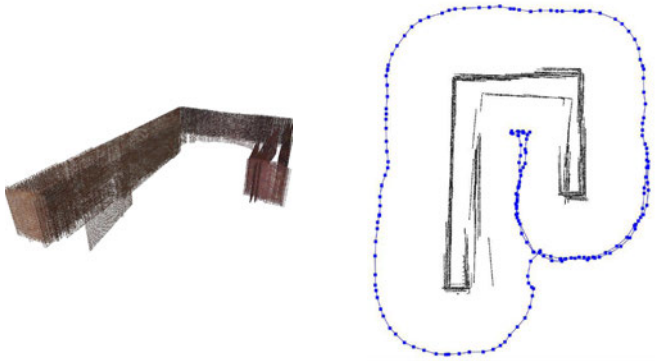


Fig. 15. (a) Large errors in localization results in poor alignment, although all portions of the structure have been captured in the model [see Fig. 16(a) for comparison]. (b) Projection of the reconstructed model on the $\mathbf{x_g y_g}$ plane, when compared with Fig. 14(a), shows the distortion introduced due to the noisy wheel odometry.

A UR5 robotic arm is used to carry the sensor, but only yaw motions of the arm are allowed, as described in Section II. For illustration purposes, we consider artificia structures made of short walllike segments. We refer to the structure used in most of the previous illustrations as the small Γ model. The large Γ model has the same shape as small Γ but is twice the size. We also illustrate the effectiveness of our policy for a realistic model of a house, and compare its performance with that of the classic frontier-based exploration (FBE) algorithm [9]. We have included a supplementary MP4 format video, which shows the simulation and real-world experiments with a Husky robot following our policies for mapping the small Γ model using a Kinect sensor.

A. Structure Size and Camera Range

The relative size of the structure with respect to the range of the camera affects the trajectory determined by our algorithms. Fig. 14 shows the simulation results for four scenarios. With a camera range of 4.5 m, the large Γ model is completely mapped at the end of the PE phase. For the small Γ model, a cavity remains, which is subsequently explored during the CE phase. Increasing the camera range to say 12 m allows the small Γ structure to be mapped at the end of the PE phase as well [see Fig. 14(b)]. One can see in Fig. 14(d) that the robot following our policies is able to map large structures with multiple cavities of different sizes. Table I lists the path lengths obtained for the different test cases.

B. Localization Accuracy

The Husky robot combines data from an inertial measurement unit (IMU), a standard GPS receiver, and wheel odometry to achieve a relatively small localization error overall. In order to evaluate the impact of localization accuracy on our algorithms, we simulate the effect of large wheel slippage by introducing a zero mean additive Gaussian white noise to each of the wheel encoder measurements, with a variance equal to $k(v_x + \omega_z)/2$, where v_x is the linear velocity of the robot, ω_z is its yaw rate, and k is a proportionality constant, also called noise level in the following. Increasing k results in a poorer alignment of the point clouds, but all portions of the structure, except the horizontal faces, are still captured in the reconstructed model (see Fig. 15). Note that our policies compute the next waypoint at discrete times and therefore assume that the drift in localization between waypoints is suff ciently small so that the robot reaches the next waypoint with the camera facing the structure.

We use the CloudCompare [42] software to compute the distortion in the reconstructed model $\mathcal{C}_k$, for a noise level k, with respect to a reference point cloud $\mathcal{C}_R$ generated using a different mobile platform with almost perfect localization. First, we register $\mathcal{C}_k$ to $\mathcal{C}_R$ using an iterative closest point algorithm [43]. We then def ne for every point in $\mathcal{C}_k$, its error to be the distance to the nearest neighbor in $\mathcal{C}_R$. Table II lists the simulation results for mapping the small Γ model with different noise levels k, where n_k is the number of points in $\mathcal{C}_k$ and μ, σ, and max are, respectively, the mean, standard

TABLE II
SIMULATION RESULTS FOR DIFFERENT LEVELS OF LOCALIZATION ACCURACY

k	n_k	μ	σ	max
0.00	65520	0.05m	0.04m	0.20m
0.25	72636	0.08m	0.06m	0.27m
0.50	82728	0.11m	0.09m	0.40m
0.75	111321	0.16m	0.13m	0.94m

deviation, and maximum value of the errors of all points in $\mathcal{C}_k$. Table II indicates that both the mean and standard deviation of the errors increase with the noise level.

C. Comparing With Frontier-Based Exploration

The frontier exploration [44] package available in ROS relies on a 2-D LIDAR to build an occupancy grid that is used to compute the frontiers. The package requires the user to def ne a 2-D polygon that encloses the structure. The algorithm then explores until there are no more frontiers inside the user-def ned polygon. In comparison with the FBE algorithm, our algorithms: 1) do not require a user def ned bounding polygon; 2) maintain as much as possible a f xed distance from the structure (during the PE phase), thereby ensuring that all portions up to a height of $H_{\max}$ are mapped; and 3) consistently explore the structure while keeping it on the right, which can be important from a user perspective to understand the behavior of the robot. On the other hand, the trajectory prescribed by the FBE algorithm depends on the size of the user-def ned bounding polygon. A large bounding polygon will cause the robot to explore areas far away from the structure and will possibly not maintain a f xed direction of exploration. Our strategy produces the same path every time for a given structure, whereas the path computed by the FBE algorithm could differ greatly between two trials. The robot also often gets stuck while using the FBE algorithm as the computed waypoints are often too close to the structure.

In order to get a quantitative measure of the structure coverage, we again use the CloudCompare software to compute essentially the projection of the reconstructed model $\mathcal{C}$ on the reference point cloud $\mathcal{C}_R$. Namely, for each point in $\mathcal{C}$, we compute the closest point in $\mathcal{C}_R$. Note that multiple points in $\mathcal{C}$ can have the same closest point in $\mathcal{C}_R$. In this case, we remove these duplicate points to obtain the *unique closest point set*. Then, as long as the reconstructed model aligns relatively well with the reference model, the cardinality of the unique closest point set is taken as our estimate of the structure coverage. Table III compares the level of structure coverage achieved by our policies and FBE with a camera range of 4.5 m for two of the environments considered. For the small Γ model, our reference point cloud has 6116 points with a minimum distance of 0.1 m between points. For the House model, our reference point cloud has 10 889 points with a minimum distance of 0.1 m between points. Since the height of the House model is more than $H_{\max}$, we only take the portion of the reconstructed model up to the height $H_{\max}$ for computing the structure coverage and mean error for the two algorithms.

TABLE III
COMPARISON BETWEEN OUR POLICY AND FRONTIER-BASED EXPLORATION

		Proposed Policy	FBE
Small Γ	Path Length	72.08m	49.78m
	Unique closest point set size	$6,063$	$5,398$
	Mean Error	0.05m	0.12m
House	Path Length	59.89m	47.55m
	Unique closest point set size	$9,182$	$7,402$
	Mean Error	0.05m	0.12m

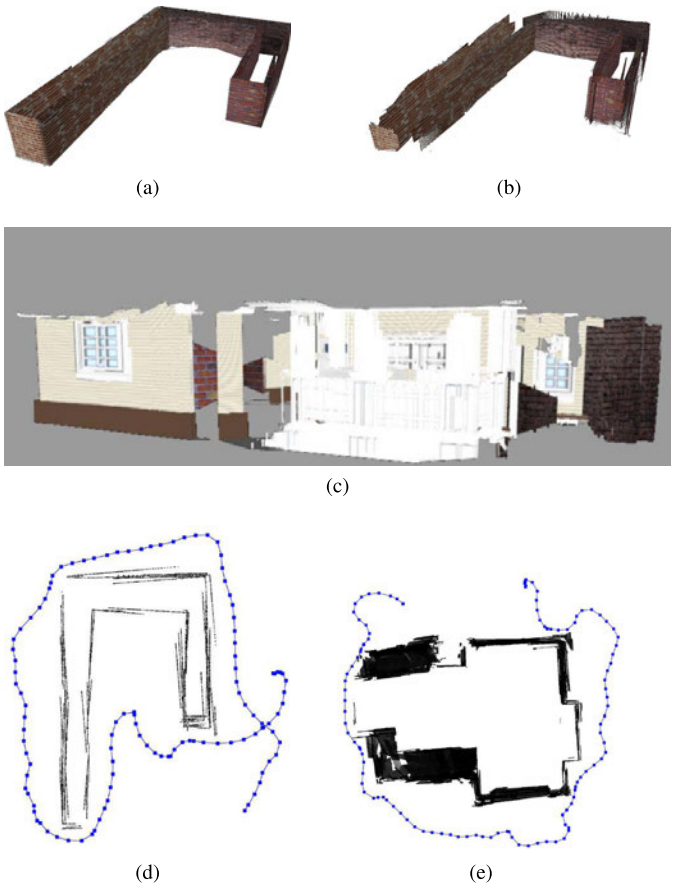


Fig. 16. (a)–(d) Comparison of the reconstructed model using our policies and FBE. Fig. (c) should be compared with Fig. 1(b). (d) and (e) Trajectories prescribed by FBE for the small Γ and House structure are shown in blue line.

Table III shows that our policies achieve a higher level of structure coverage than FBE for the environments considered and our proposed coverage metric. Note also that the smooth trajectory prescribed by our policies is benef cial to the vSLAM module to achieve a better alignment and a lower value for the mean error in the reconstructed model, especially if the robot localization accuracy is poor. A visual inspection of Fig. 16(a) and (b) shows the improvement in model reconstruction when using our algorithms compared with FBE.

Fig. 17. Left: Husky robot used in our experiments with the robotic arm, depth sensor, and laser range scanner. All other sensors seen are not used. Right: indoor structure being inspected.

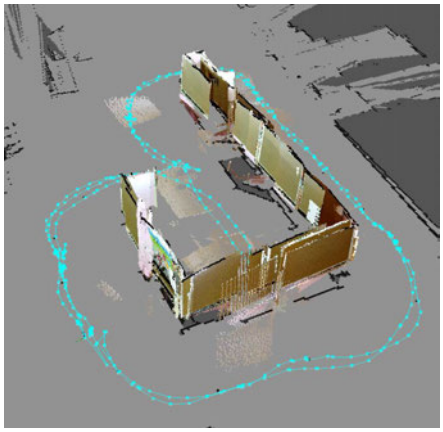


Fig. 18. Angled view of the constructed model. The cyan lines with squares indicate the path taken by the robot.

VII. Real-World Experiment

Here, we discuss an example of real-world experiment using the Husky robot shown in Fig. 17, equipped with a Kinova robotic arm carrying a Kinect v2 depth sensor. An SICK laser range scanner is used only for obstacle detection. All computations are done in real time on an embedded Intel i5-based computer without the help of a dedicated GPU. The structure was built from cork display panels and foam insulation boards. As shown in Fig. 17, it mimics the small Γ model used in simulation and its dimensions are 8.2 m $\times$ 4m. To help the vSLAM algorithm detect a loop closure, we place a visual marker (a colorful poster seen in Fig. 17) on the structure in front of the starting point of the robot. Since the panels are similar on both sides, we also put visual markers inside the structure to confirm that the cavity inspection has correctly mapped all of the inside. The most computationally intensive part of our algorithm is the detection of cavity entrances, (see Section IV-A) which takes a few seconds of computation for this structure at the end of the PE phase.

Due to the limited space available to maneuver around the structure, we reduce the obstacle sensing region to a range of 1.8 m and to a 10° cone in front of the robot. The depth camera range is cut at 4 m and we set the desired wall distance D to 1.5 m. For odometry, we use the extended Kalman filter (EKF) from [45] with IMU and wheel odometry data as inputs. The output of the EKF is sent to RTAB-Map [6] for mapping and localization purposes. Since we use a ground robot on flat terrain, we constrain RTAB-Map's mapping to three degrees of freedom (x, y and yaw angle).

The behavior of our algorithm is illustrated on the accompanying video, and the model produced online with the vSLAM module is shown in Fig. 18. Overall, the executed trajectory confirms that our algorithm correctly performs PE followed by cavity inspection. Noise in the depth measurements can be an issue if left unfiltered. However, proper calibration and applying a standard speckle and bilateral filter can alleviate the problem. In our experiments, we tuned the parameters of these filters by placing the robot in a location where significant noise was measured. We then increased the maximum speckle size and size of the bilateral filter window until most of the visible noise was removed.

The experiment also illustrates some practical issues that can degrade mapping performance. First, because of the height of the boards used to build the structure and the limited space available to navigate around it, the robotic arm was extended so that the depth sensor was at a height of 1.2 m. It then tended to shake during acceleration changes of the robot, making the sensor vulnerable to producing blurry images. This could be mitigated by a better control of the smoothness of the robot trajectories, a stiffer orientable platform to hold the sensor, and by using a stereo camera with global shutters in bright daylight to reduce motion blur. During testing, we also noticed that the algorithm can be sensitive to gaps or "windows" in the structure. In front of a gap, the depth sensor can detect surfaces inside the structure, which can then perturb the goal calculation algorithm described in Section III-A. This can be addressed by reducing the range of the sensor measurements to a value close to the desired distance D.

In summary, among the possible failure cases of our algorithm, we identified during our experiments and simulations: 1) waypoint determination errors due to the presence of gaps or windows in the structure and 2) incorrect global loop closures due to the possible self-similarity of the structure, which can be mitigated by adding a unique marker at the starting point.

VIII. Conclusion

This paper presents motion planning strategies that guide a mobile ground robot carrying a camera or depth sensor to autonomously explore the visible portion of a bounded 3-D structure. The proposed policies do not assume any prior information about the size or geometry of the structure. Coupled with state-of-the-art vSLAM systems, our strategies are able to achieve high coverage in the reconstructed model, given the physical limitations of the platform. We illustrate the efficacy of our approach via 3-D simulations for different structure sizes, camera range, and localization accuracy, and we have tested our system in real-world experiments. In addition, a comparison of our policies with the classical frontier-based exploration algorithms clearly shows the improvement in performance for a realistic structure such as a house.

REFERENCES

[1] M. Jacobi, "Autonomous inspection of underwater structures," *Robot. Auto. Syst.*, vol. 67, pp. 80–86, May 2015.

[2] S. F. El-Hakim, J. A. Beraldin, M. Picard, and G. Godin, "Detailed 3D reconstruction of large-scale heritage sites with integrated techniques," *IEEE Comput. Graph. Appl.*, vol. 24, no. 3, pp. 21–29, May 2004.

[3] Y. Turkan, F. Bosche, C. T. Haas, and R. Haas, "Automated progress tracking using 4D schedule and 3D sensing technologies," *Autom. Construction*, vol. 22, pp. 414–421, Mar. 2012.

[4] I. Brilakis, H. Fathi, and A. Rashidi, "Progressive 3D reconstruction of infrastructure with videogrammetry," *Autom. Construction*, vol. 20, no. 7, pp. 884–895, Nov. 2011.

[5] J. J. Lin, K. K. Han, and M. Golparvar-Fard, "A framework for model-driven acquisition and analytics of visual data using UAVs for automated construction progress monitoring," in *Proc. Int. Workshop Comput. Civil Eng.*, Austin, TX, USA, Jun. 2015, pp. 156–164.

[6] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Chicago, IL, USA, Sep. 2014, pp. 2661–2666.

[7] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.

[8] R. Bajcsy, "Active perception," *Proc. IEEE*, vol. 76, no. 8, pp. 966–1005, Aug. 1988.

[9] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. (CIRA)*, Washington, DC, USA, Jul. 1997, pp. 146–151.

[10] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficien next-best-scan planning for autonomous 3D surface reconstruction of unknown objects," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 611–631, Dec. 2015.

[11] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 5031–5037.

[12] A. Bircher *et al.*, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 6423–6430.

[13] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," in *Proc. Int. Conf. Automat. Planning Scheduling (ICAPS)*, São Paulo, Brazil, Jun. 2012, pp. 29–37.

[14] L. Yoder and S. Scherer, "Autonomous exploration for infrastructure modeling with a micro aerial vehicle," in *Field and Service Robotics*. Springer, 2016, pp. 427–440.

[15] W. Sheng, H. Chen, and N. Xi, "Navigating a miniature crawler robot for engineered structure inspection," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 2, pp. 368–373, Apr. 2008.

[16] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Auto. Robots*, vol. 3, no. 2, pp. 91–119, Jun. 1996.

[17] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 331–344, 2002.

[18] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 345–366, 2002.

[19] R. S. Lim, H. M. La, and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 367–378, Apr. 2014.

[20] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from Internet photo collections," *Int. J. Comput. Vis.*, vol. 80, no. 2, pp. 189–210, 2007.

[21] J.-M. Frahm *et al.*, "Building Rome on a cloudless day," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 368–381.

[22] C. Wu. (2011). *VisualSFM: A Visual Structure from Motion System*. Accessed: Dec. 27, 2015. [Online]. Available: http://ccwu.me/vsfm/

[23] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[24] S. Daftry, C. Hoppe, and H. Bischof, "Building with drones: Accurate 3D facade reconstruction using MAVs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 3487–3494.

[25] K. Tuite, N. Snavely, D.-Y. Hsiao, N. Tabing, and Z. Popovic, "PhotoCity: Training experts at large-scale image acquisition through a competitive game," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 1383–1392. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979146

[26] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 3607–3613.

[27] *Gazebo Robot Simulator*. Accessed: Dec. 29, 2015. [Online]. Available: http://gazebosim.org/

[28] R. Shade and P. Newman, "Choosing where to go: Complete 3D exploration with stereo," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 2806–2811.

[29] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2012, pp. 9–15.

[30] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficien visual exploration and coverage with a micro aerial vehicle in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 1071–1078.

[31] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 4775–4782.

[32] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An eff cient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[33] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, "Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain," *J. Field Robot.*, vol. 29, no. 2, pp. 277–297, Mar./Apr. 2012.

[34] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Adv. Robot.*, vol. 27, no. 6, pp. 459–468, 2013.

[35] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 1–4.

[36] N. J. Mitra, A. Nguyen, and L. Guibas, "Estimating surface normals in noisy point cloud data," *Int. J. Comput. Geometry Appl.*, vol. 14, no. 04n05, pp. 261–276, 2004.

[37] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[38] H. Choset *et al.*, *Principles of Robot Motion—Theory, Algorithms and Implementation*. Cambridge, MA, USA: MIT Press, Jun. 2005.

[39] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Syst. Mag.*, vol. 28, no. 3, pp. 36–73, Jun. 2008.

[40] *ROS Navigation Stack*. Accessed: Nov. 26, 2015. [Online]. Available: http://wiki.ros.org/navigation

[41] *Gazebo Plugins*. Accessed: Dec. 22, 2015. [Online]. Available: http://wiki.ros.org/gazebo_plugins

[42] (2016). *CloudCompare (Version 2.6.0) [GPL Software]*. [Online]. Available: http://www.cloudcompare.org/

[43] S. Rusinkiewicz and M. Levoy, "Efficien variants of the ICP algorithm," in *Proc. 3rd Int. Conf. 3-D Digit. Imag. Modeling*, Quebec City, QC, Canada, May 2001, pp. 145–152.

[44] *ROS Frontier Exploration*. Accessed: Nov. 26, 2015. [Online]. Available: http://wiki.ros.org/frontier_exploration

[45] T. Moore and D. Stouch, "A generalized extended Kalman f lter implementation for the robot operating system," in *Proc. 13th Int. Conf. Intell. Auto. Syst. (IAS)*, Jul. 2014, pp. 335–348.

Manikandasriram Srinivasan Ramanagopal received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2016. He is currently pursuing the Ph.D. degree in robotics with the Robotics Institute, University of Michigan, Ann Arbor, MI, USA.

His research interests include the intersection of perception and control for autonomous vehicles.

André Phu-Van Nguyen received the B.Eng. degree in computer engineering from Polytechnique Montreal, Montreal, QC, Canada, in 2015, where he is currently pursuing the M.Sc.A. degree in electrical engineering.

His research interests include revolve around making mobile robots smart and autonomous.

Jerome Le Ny (S'05–M'09–SM'16) received the B.S. degree in engineering from the École Polytechnique, Palaiseau, France, in 2001, the M.Sc. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2003, and the Ph.D. degree in aeronautics and astronautics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008.

From 2008 to 2012, he was a Post-Doctoral Researcher with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA. He is currently an Associate Professor with the Department of Electrical Engineering, Polytechnique Montreal, Montreal, QC, Canada, and a member of GERAD, Montreal, QC, Canada, a multiuniversity research center on decision analysis. His research interests include planning under uncertainty, robust and stochastic control, and networked control systems, with applications to autonomous multirobot systems and intelligent infrastructures.